Interactive Real-time Multimedia Applications on Service Oriented Infrastructures

ICT FP7-214777

QoS Aware Intelligent Storage White Paper

QoS_Aware_Intelligent_Storage_White_Paper_v1_0

Delivery: 14.01.2011
Version: 1.0

Project co-funded by the European Commission within the 7th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Responsible Partner: XYRATEX (XY)

Revision history:

Date	Editor	Status	Version	Changes
28/01/2011	**Sai Narasimhamurthy**	**Draft**	**0.1**	**Initial Draft**
31/01/2011	**Sai Narasimhamurthy**	**Draft**	**0.2**	**First set of changes**
03/02/2011	**Sai Narasimhamurthy**	**Draft**	**0.3**	**Second set of changes**
04/02/2011	**Sai Narasimhamurthy**	**Draft**	**0.4**	**Internal QA Version**
13/02/2011	**Sai Narasimhamurthy**	**Final**	**1.0**	**Final version after internal QA**

Authors

Sai Narasimhamurthy (XY)
Ganesan Umanesan (XY)
James Morse (XY)
Malcolm Muggeridge (XY)

Internal Reviewers

Wolfgang Huther (Precision Mechatronics), Roland Kuebert (USTUTT) and Spyridon Gogouvitis (NTUA)

Acknowledgements

The research leading to these results has received funding from the EC Seventh Framework Programme FP7/2007-2011 under grant agreement n° 214777

More information

The most recent version of the public deliverables of IRMOS can be found at
http://www.irmosproject.eu

Glossary of Acronyms

Acronym	Definition
ASC	Application Service Component
A-SLA	Application Service Level Agreement
EE	Execution Environment
IaaS	Infrastructure as a Service
I/O	Input/Output
IRMOS	Interactive Real-time Multimedia Applications on Service Oriented Infrastructures
ISONI	Intelligent Service Oriented Network Infrastructure
KVM	Kernel Virtual Machine
LTS	Long Term Storage
MDS	Meta Data Server
MDT	Meta Data Target
OSS	Object Storage Server
OST	Object Storage Target
PaaS	Platform as a Service
QoS	Quality of Service
RAID	Redundant Array of Independent Disks
SLA	Service Level Agreement
T-SLA	Technical Service Level Agreement
VM	Virtual Machine
VMU	Virtual Machine Unit
VSN	Virtual Service Network

Table of Contents

List of Figures

Executive Summary

The IRMOS project envisions Quality of Service (QoS) capabilities for Soft Real-time applications in a cloud based service environment. Providing application QoS in such an environment entails stringent performance requirements on all the individual "tiers" of the cloud environment. The Infrastructure as a Service tier, the Platform as a Service tier and the Application as a Service (or, Software as a Service) tier should all be QoS aware and work seamlessly with each other to guarantee application QoS. The storage subsystem forms a fundamental part of the Infrastructure as a Service tier for which QoS capabilities, and precise details of the interworking of the subsystem with the other elements of the cloud, without "breaking" perceived application QoS, needs to be provided. This White Paper focuses on these aspects and highlights the key innovations in storage for cloud computing advanced by the work within the context of the IRMOS project.

Index Terms: Quality of Service, Infrastructure as a Service, Platform as a Service, Application as a Service, Software as a Service, Cloud environment, Soft Real-time, cloud computing, data storage

1. Introduction

The emergence of cloud computing in the last decade has begun the transition of in-house IT resources and applications in enterprises and businesses to cloud platforms hosted by third parties. Cloud computing has caused a significant reduction in IT budgets because it is no longer necessary to over-provision IT resources and applications to cater to worst case scenarios. The applications and the IT infrastructure can now be accessed "on demand", as a utility allowing the provisioned service to fluctuate based on actual demands at various times, leading to potential cost savings.

The concept of "Real-time" applications has existed in the realm of computing for a long time, in which applications provide very precise response times and failure to do so may lead to catastrophic system failures. Applications that are used in embedded systems within aircrafts, air traffic control, etc are good examples. There is also a class of "Soft" Real-time applications which do require predictable response times, but where variation in response does not produce such systemic breakdowns. Such a class of applications are proliferating with the emergence of cloud computing. Many multimedia applications such as Massively Multiplayer Online Gaming and high-quality internet video conferencing applications reasonably fall under this category.

As of today, the cloud is inherently not "Real-time", or even, "Soft Real-time" capable. This capability can only be introduced by making all the "tiers" of the cloud platform (the application, platform and infrastructure tiers) Quality of Service aware. Quality of Service awareness in this case is the ability to promise a given level of service for a customer, in terms of application response times, throughputs and some system requirements (such as available storage capacity). These QoS requirements trickle down as infrastructure tier requirements, such as latency, bandwidth and jitter requirements for each of the computing, networking and storage elements. Complex QoS aware cloud platforms were non-existent, until recent research initiatives within the government, academia and industry R&D groups. The IRMOS project [1] envisions such an end-to-end Soft Real-time cloud environment for the new class of aforementioned applications.

The QoS Aware Intelligent Storage platform which was developed under the context of the project, and which we will discuss in this White Paper, forms a fundamental part of the Soft Real-time infrastructure tier of the IRMOS cloud.

We next discuss the IRMOS cloud before we delve into the details of how the QoS awareness and associated QoS guarantees within storage was achieved.

1.1. The IRMOS Cloud

The IRMOS cloud environment is essentially an application development framework, deployment platform, and a methodology for implementing Soft Real-time awareness in clouds [5]. The IRMOS cloud consists of an Application tier, a Platform tier and an Infrastructure tier. The Application tier in IRMOS specifies methods to develop applications for IRMOS, the Platform tier (called Framework Services) provides a means of discovery and reservation for infrastructure resources as well as monitoring and management functions for applications. The Infrastructure tier called ISONI (Intelligent Service Oriented Networking Infrastructure) consists of the network fabric, storage and processing elements for the running applications.

As dictated by the norms of Service Oriented Architectures (SOAs), the key concept in IRMOS is to split up a Soft Real-time application into independent, but interconnected "Application Service Components" (ASCs), executing independently within a cloud based infrastructure and ultimately working together providing QoS to the application's end users. The independent ASCs which usually execute in their own compute environments (E.g: Virtual machines in a physical host anywhere in the cloud) are logically connected through abstractions called Virtual Service Networks (VSNs). An executing application in IRMOS thus corresponds to a VSN.

1.2. Storage Service Level Agreements

The users of IRMOS applications specify the QoS requirements through Service Level Agreements (SLAs). At the application tier, the SLAs consist of high level application parameters as needed by the IRMOS user. Users, for example, may specify the application's response time or similar high level requirements. Framework Services translates these high level SLA requirements or A-SLAs (Application Service Level Agreements) to low level SLAs as required by the infrastructure called T-SLAs or Technical SLAs. The T-SLA parameters corresponding to the storage (or a "storage pool" in IRMOS parlance), which caters to an application's data storage consists of the following:

a. Capacity: "How large a storage pool needs to be reserved for a running application?"
b. Lifetime: "How long is the storage pool, or a connection to the storage pool, needed for a running application?"
c. Bandwidth: "What bandwidth needs to be supported by the storage pool, or by a connection to the storage pool?"
d. Latency: "What latency needs to be supported by the storage pool, or by a connection to the storage pool?"
e. Resiliency: "What reliability metrics need to be supported by the storage pool, primarily, RAID levels?"
f. Jitter: "What jitter, or variability in latency can be tolerated by connections to the storage pool?"

Answers to these questions form binding agreements between storage in ISONI and the higher tiers in the IRMOS cloud, ultimately leading to A-SLA adherences.

We discuss a framework to provide these storage T-SLA guarantees as well as the "science" behind how these decisions are made, before progressing to briefly explain the key innovations of the work.

2. Storage QoS Framework

2.1. Overview

Without going into details of the rest of the modules in the IRMOS application, platform and infrastructure tiers, (which can be obtained through the IRMOS public deliverables in [1]), we deal directly with the pertinent storage modules in the QoS aware storage framework in ISONI. The storage QoS framework in IRMOS consists primarily of:

1. Storage Manager
2. Repository Manager
3. A parallel file system[1] and associated storage hardware

These elements interact and work together to guarantee Quality of Service as described by the storage specific T-SLAs. The depiction in Figure 1 is the Storage QoS framework in an ISONI "Node", with every Node having its own dedicated storage. Many Nodes are hierarchically organised as a "Domain".

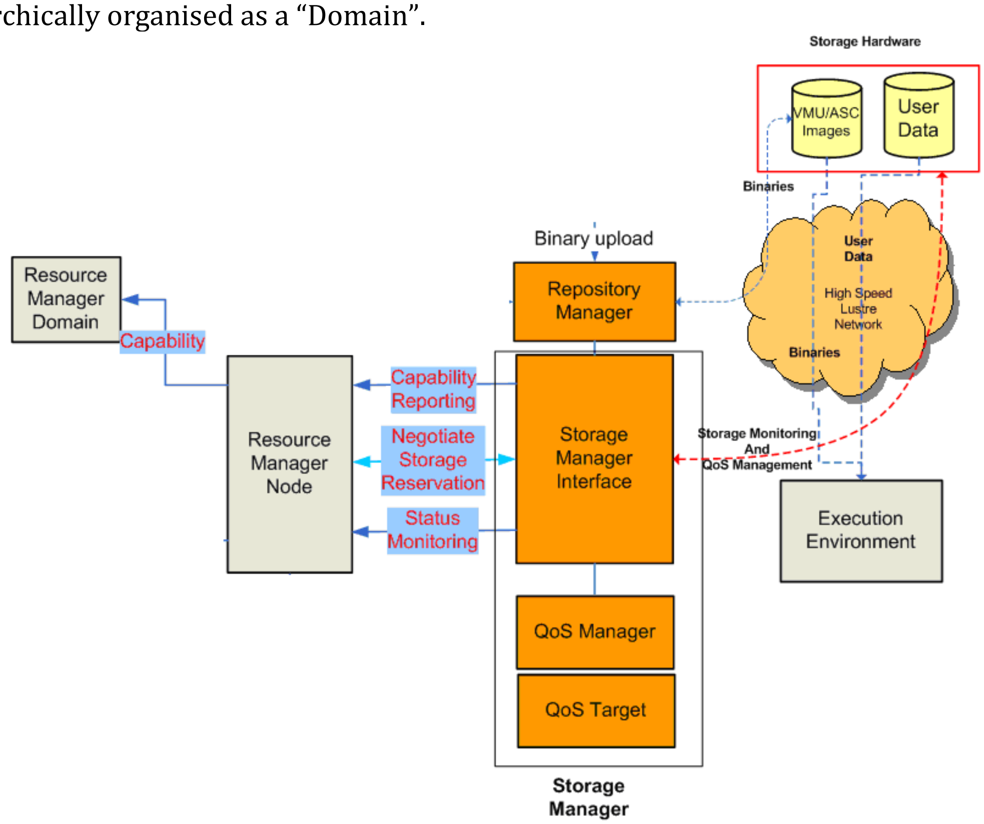

Figure 1 Storage QoS framework in IRMOS ISONI

[1] We use the Lustre file system [2] since its open source and well proven in the high performance computing industry.

The storage specific T-SLA parameters are passed to a "Resource Manager" module in ISONI which in turn passes them to the Node level Storage Manager ("Negotiate Storage Reservation" functionality of the Storage Manager Interface in Figure 1). These T-SLAs consist of storage parameters as discussed in section 1.2. The Storage Manager then utilizes the functionalities of the QoS Manager and the QoS Target to make decisions on accepting or rejecting these SLAs. These decisions are made based on the feasibility of the storage pool to accommodate these requirements. These feasibility decisions are made through the "System Capability" based methodology which we will discuss in the next section.

The actual storage pools which host application data are implemented using a Lustre parallel file system based storage network in every ISONI Node. The ASCs run on virtual machines ("Execution Environment" in figure 1) on Physical Hosts that contain Lustre client software modules and access storage in the Lustre Object Storage Servers (OSSs) which in turn contain the actual data in their Object Storage Targets (OSTs). The clients and servers are typically connected through a high speed Infiniband fabric. The QoS Manager periodically queries the storage pool through the Storage Manager Interface to determine System Capability (discussed in detail in section 3). The Storage Manager also provides monitoring and capability reporting functionality to the Framework Services through the Resource Manager.

ASCs are uploaded as binaries onto ISONI storage before their execution, through a Repository Manager module. These are brought into the Execution Environment before the ASCs start execution.

2.2. Storage QoS Framework Implementation

The Storage Manager modules and the Repository Manager can be implemented using the servers which are part of the Lustre parallel file system.

In the current implementation of the Storage QoS framework, the Storage Manager Interfaces are hosted on the same physical machine that hosts the Lustre Metadata Server, which has a "centralized" view of Lustre file system data. The QoS Manager is also hosted on this same machine. The QoS Target portion of the Storage Manager framework is implemented in the Lustre Object Storage Servers with modified standard Lustre server-side software. The Lustre clients are physical machines (with modified standard Lustre client-side software) which contain Virtual Machines running ASCs that need access to storage pools adhering to previously discussed storage specific T-SLAs. The KVM hypervisor [3] is used in the Lustre client machines to achieve the virtualisation functionality. Figure 2 depicts the Storage QoS framework implementation as it relates to the Lustre file system.

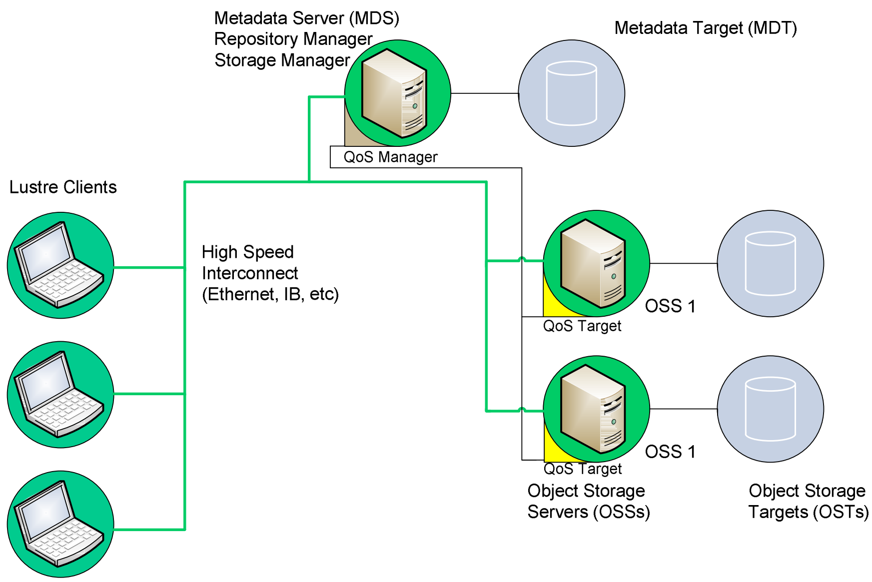

Figure 2 Storage QoS Framework Implementation

3. Storage QoS Management

3.1. Overview

Storage QoS Management requires two key abilities on the part of the Storage QoS Framework, which are not available in today's standard storage solutions:

1. The ability to make decisions regarding whether to accept or reject SLA requests from ASCs (SLA based, QoS Aware Decision Making)

2. Once the decision to accept the SLA has been made, there is a need to guarantee the given performance for the ASCs (SLA based QoS Guarantees)

There is a need to build performance models for the storage pools within the Lustre file system, which would help to accurately perform SLA based decision making. The SLA requests arrive at the Storage Manager which makes decisions on accepting or rejecting the SLA request.

The QoS Manager maintains a performance model of the storage pool(s) obtained through highly detailed performance profiling of the storage subsystem. These performance profiles translate to what is termed "System Capability" for the storage pool and are used for SLA based decision making.

After an SLA is accepted, the SLA parameter(s) for the ASCs (for example, a request for a given bandwidth) are translated to "weights" which are then assigned by the QoS Target to the respective ASC's connection to storage at the Lustre Object Storage Server, through a "Request Scheduler" (RS). These assigned weights assure that the performance guarantees are constantly met, and are unhindered by any other "rogue" or interfering applications accessing the same storage hardware.

Figure 3 depicts Storage QoS Management by the Storage Manager, through the QoS Manager. I/O requests from a user (ASC in this case) are now weighted according to their SLAs and are serviced by the Request Scheduler. These "controlled" I/Os ultimately find their way to the storage hardware.

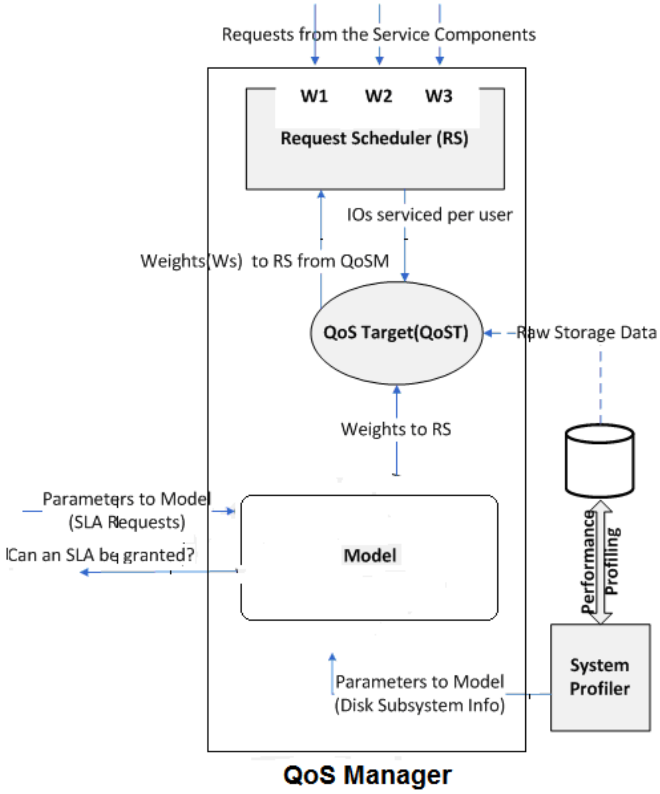

Figure 3 Storage QoS Management by the QoS Manager

If any of the SLA parameters cannot be satisfied, the SLA is rejected and the Storage Manager conveys the information to the other ISONI modules through the Resource Manager Node. The application tier is informed of the infeasibility of meeting the SLA and the customer takes appropriate action, such as re-negotiating new SLAs. The storage pools are continuously monitored, which enables addressing SLA violations from the storage infrastructure, through informing the higher tiers, which then take appropriate action on behalf of the IRMOS customers.

3.2. SLA based QoS Aware Decision Making

The storage subsystem (storage hardware along with the associated software components) has certain fundamental limits on the performance that it can provide for applications. These fundamental limitations are characterised by storage "System Capabilities". There are limits to various SLA parameters, namely, capacity, bandwidth, latency and jitter. The limitation on capacity is very easy to understand. Storage capacity indicated by the storage hardware manufacturer can indeed be thought of like a circular pie which can be precisely distributed amongst the various competing constituents/applications (or ASCs, in our case). However, we question if such an analogy holds true for parameters such as bandwidth.

There are limitations on bandwidth, latency, etc that are indicated by the manufacturers for certain hardware components in the storage subsystem (Eg: Specification of sustained data rates for disks). However, a complex storage subsystem such as that based on Lustre consists of many hardware and software components such as disk caches, networking software, device drivers and file system caches, just to name a few. How then to understand the performance limitations, where the individual limitations of each of these sub-components are no longer advertised or indicated? Further, the "sum total" of the limitations of these components cannot be comprehended as simple mathematical summations of the limitations of the sub-components since such an analogy severely discounts the complex interactions that underlie the combined sub system.

Hence the fundamental limitations such as bandwidth and latency for complex subsystems can only be understood by detailed performance profiling and then mathematical modelling of the storage system as a whole. This forms the essence of "System Capability". There can hence be "Storage Bandwidth Capability" and "Storage Latency Capability" of a storage system that represents the ultimate limitations on these parameters under a given set of application workloads (Block Size, the randomness in I/O access or "Sequentiality", number of applications accessing the storage pool, mix of reads/writes, etc).

Thus given the application workload characteristics of an ASC, the System Capability of the storage system can be estimated through such a mathematical model.

In the process of arriving at mathematical models, an in depth study of the storage subsystem performance for various block sizes, randomness of access (termed sequentialities), read/write mix and the number of client connections is performed as an offline process within an IRMOS test bench. Figure 4 exemplifies the probability distribution profile of latencies for a given Block size and Sequentiality, which is used in mathematical modelling. Figure 5 depicts the "heat maps" denoting latency responses of the storage system – with every storage system having unique heat maps, which are ultimately used in developing performance models.

Figure 6[4] denotes an example System Capability ("mu", which is a measure of Bandwidth in MB/s), which is represented for various Block Sizes and Sequentialities for 100% Read access to a storage pool. Figure 6 is a result of understanding probability distributions of performance metrics. These are finally used to develop mathematical models for the storage system, which predicts performance for any given set of ASC/application workload parameters. Thus predicted latency and/or bandwidth for an application with, say, 16KB block sizes, 65% sequentially accessed , 100% Read workload and being the 4^{th} connection to the storage pool, can be predicted through the System Capability model.

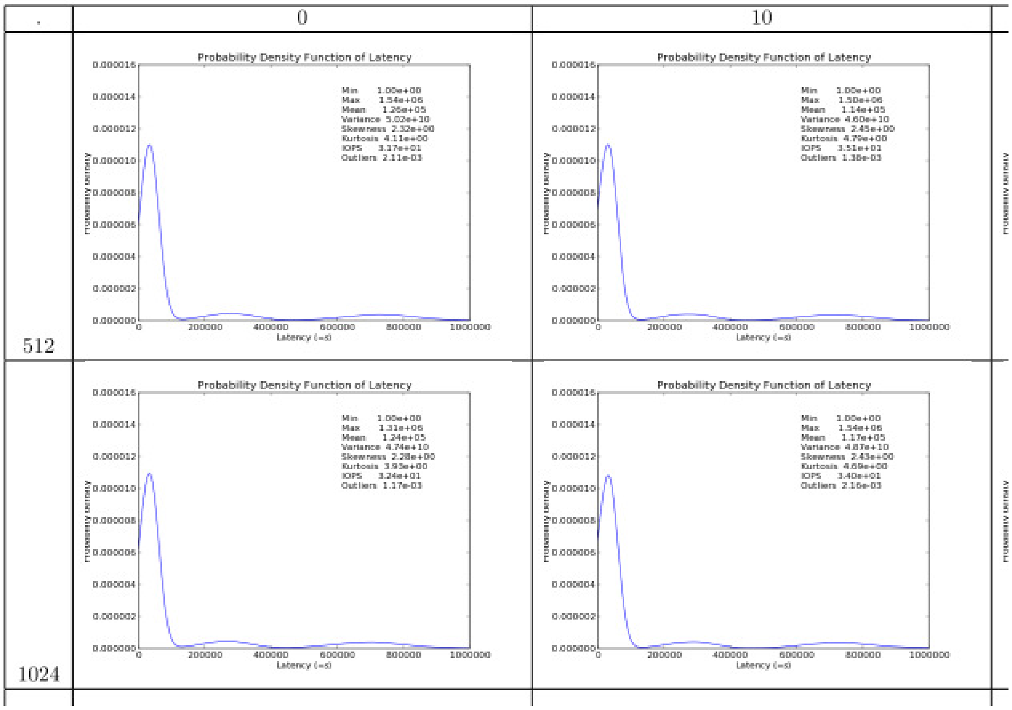

Figure 4 Example performance profiles for a storage pool used to arrive at System Capability (1) (The storage pool is part of the IRMOS test bench)

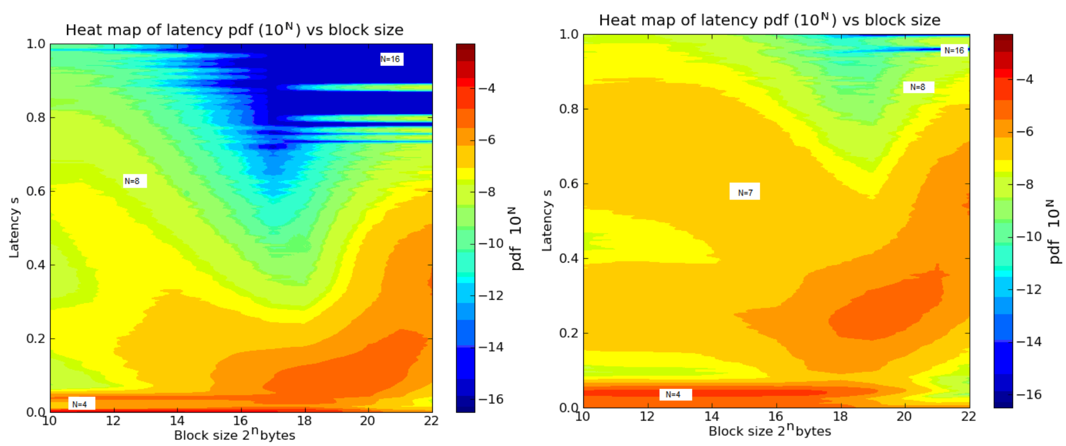

Figure 5 Example performance profiles for a storage pool used to arrive at System Capability (2). (Left heat map is for sequentially accessed data for a Lustre storage system and the right heat map is for randomly accessed data for a Lustre storage system. The Lustre storage system is part of the IRMOS test bench)

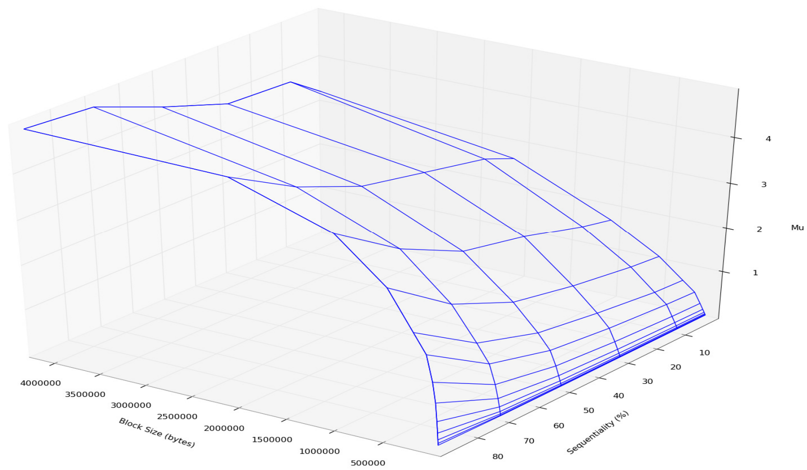

Figure 6 Example System Capability plot used for deriving the mathematical model for predicted performance (For a storage pool in the IRMOS test bench) [4]

Once the mathematical model for the predicted performance of an ASC workload is known, SLAs can be easily accepted or rejected by the QoS Manager. For example, if the requested performance is greater than predicted available performance, than the SLA is rejected.

3.3. SLA based QoS Guarantees

Once the SLA parameters corresponding to an ASC are granted, performance needs to be sustained at the required levels (as defined by the SLA) irrespective of the presence of potential rogue applications within the system. As discussed earlier, SLA parameters such as capacity are easy to guarantee, unlike other parameters such as bandwidth and latency.

For example, an ASC with an SLA to receive 40 MB/s should continue to receive 40 MB/s regardless of the actions of other ASCs or rogue applications. This "guarantees" the net bandwidth for an application, consisting of multiple ASCs.

As indicated earlier, algorithms are implemented within the QoS Manager and the QoS Target that intelligently assigns weights to ASCs based on their SLAs so that the performance, in effect, is "throttled" to what has been requested. These weights are all maintained by a Request Scheduler within the Lustre Object Storage Server in IRMOS ISONI.

Figure 7 shows the example of SLA based QoS Guarantees accomplished in the IRMOS test bench. We denote 4 ASCs corresponding to an application by App1, App2, App3 and App4. All the ASCs are initially "unthrottled", in that, they all use the available System Bandwidth Capability of ~400 MB/s in a best effort fashion. They are vulnerable to interference from rogue applications (and each other). The SLAs are then established for the ASCs for 10MB/s, 30MB/s, 50MB/s and 75MB/s, as these values are well within the System Bandwidth Capability for the storage pool, for a given time period. After some time, the scheduler is disabled. New SLAs are then negotiated with new bandwidth values for these ASCs, and the scheduler is enabled again. Successive bandwidth throttling and unthrottling is thus denoted until the time 16:11:37 in the figure.

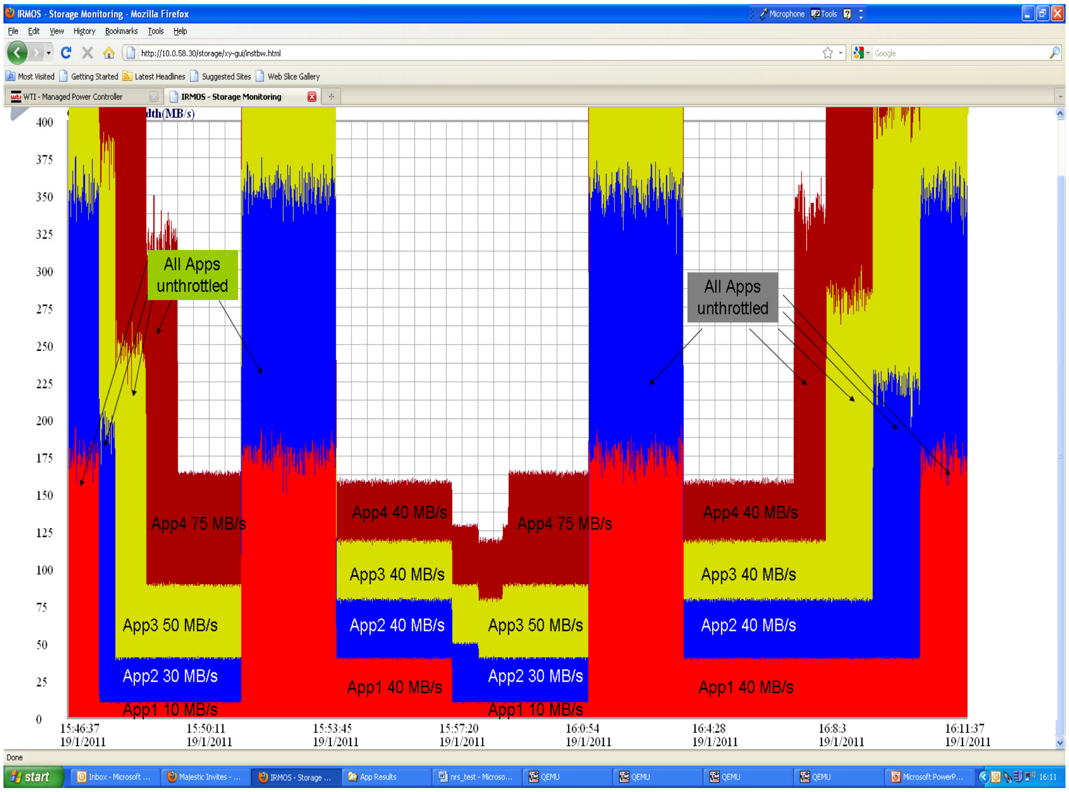

Figure 7 SLA based Performance Guaranteeing for a storage pool in the IRMOS test bench

Bandwidth performance monitoring depicted above is done through specialised monitoring tools developed as part of IRMOS ISONI Storage QoS Framework.

4. Key Innovations

The following were the key innovations brought forth by the work in the larger context of cloud based applications.

1. *Storage Intelligence*: Through the Storage Manager framework, QoS aware storage is demonstrated which is "intelligent" enough to identify its own capabilities (capacity, bandwidth, latency, etc) through performance modelling, and report it to higher tiers in a cloud environment.

2. *Server Side Storage QoS guarantees:* The work demonstrates precise performance throttling of applications from the Storage Server Side. Performance is guaranteed for an application/ASC even in the presence of other applications which can potentially "steal" storage resources such as bandwidth.

3. *Advanced Storage Interfaces:* Through the Storage Manager interface, the QoS guaranteed storage platform enables applications distributed over a cloud to negotiate QoS parameters such as capacity, bandwidth and latency. Aided by future standards bodies, this interface could further be developed to interoperate with any service oriented cloud based environment.

4. *Advanced Storage Monitoring:* Through the storage monitoring implemented within the Storage Manager, a method to continuously monitor the storage system in a service oriented cloud (for advanced SLA parameters such as bandwidth and latency and not just capacity, which legacy systems currently provide) has been developed.

5. Conclusion

The key innovations achieved in the process of developing the QoS Aware Intelligent storage framework through the Storage Manager feature in IRMOS successfully demonstrated SLA based negotiation, System Capability based decision making and SLA based performance guarantees – which will all be key requirements for the deployment of next generation Soft Real-time applications in clouds. These innovations provide new, much needed storage management capabilities in other more traditional data processing systems. Storage QoS is also the key piece ultimately in the realisation of end-end application QoS awareness/guarantees in clouds, whose vision is now close to fulfilment through IRMOS.

6. References

[1] Online, Retrieved January 2011, http://www.irmosproject.eu/
[2] Online, Retrieved March 2010, Lustre White Paper, http://www.sun.com/offers/details/lustre_networking. html
[3] Online, Retrieved January 2011, http://www.linux-kvm.org/page/Main_Page
[4] S.Narasimhamurthy, J.Morse, D.Golbourn, G.Umanesan and M.Muggeridge, "Storage Quality of Service for High Performance Compute Clouds", UK e-Science All Hands Meeting 2010, September 2010
[5] Online, Retrieved January 2011, http://www.irmosproject.eu/Deliverables/Default.aspx, WP3_3_1_2_v1_0 IRMOS Overall Architecture

